**STL 과제1 보고서**

**2013184043 게임공학부 양태윤**

**1. 과제 해결방법**

수업시간에 했던 내용을 사용하여 코드를 작성하였습니다. 자료구조는 바이너리 형태로 파일 입출력하기 쉽고 순차적으로 하는 일이 많기 때문에 벡터를 사용했습니다.

**1-1.RankinData 클래스**

플레이어 정보를 담는 클래스입니다. 플레이어 ID, 현재 순위, 과거 순위, 현재 상위 몇 퍼센트에 위치하는지, 쿠키훈련소 점수, 떼탈출 점수를 맴버변수로 갖고 있습니다. 각 멤버변수는 private으로 감춰뒀고 인라인으로 get/set함수를 만들어줬습니다. get함수는

함수 내부에서 값이 바뀔일이 없으므로 함수선언뒤에 const를 적었고 set함수는 받아오는 인자에 레퍼런스와 const를 썼습니다. 기본생성자는 없으면 파일입출력 시 resize함수를 쓸 수 없어서 만들어줬습니다. 파일출력이 메모리를 통째로 쓰는 방식이기 때문에 파일입력받을 때도 미리 벡터의 크기를 잡아줘서 메모리를 통쨰로 받아올 필요가 있었습니다. 다른 생성자로는 플레이어 초기화를 위한 각 멤버변수에 들어갈 값을 인자로 받는 생성자가 있습니다. 이외의 멤버함수로는 멤버변수들을 출력해주는Print함수가 있습니다. 출력해보니 점수의 자릿수가 다양해 열이 안 맞는 문제가 있었습니다. 그래서 출력자릿수를 고정하는 함수 setw()를 썼습니다. 상위 몇프로인지 나타내는 ratio 변수도 그대로 출력하면 너무 숫자가 길어지기 때문에 precision 함수를 써서 출력되는 소수점자릿수를 3번째 자리에서 반올림하였습니다.

**1-2.main문**

몇명 플레이 할건지는 상수로 지정했습니다. 변수로는 점수를 구하기 위한 노말 디스트리뷰션 변수와 플레이할 10000명을 구하기 위한 유니폼 디스트리뷰션 변수가 있습니다. RankingData를 템플릿으로 가지는 벡터변수가 있습니다. 렌덤엔진변수의 시드값은 랜덤디바이스를 이용해서 넣어줬습니다. ifstream클래스로 파일입력을 처리했습니다. 플래그는 바이너리로 주었고 파일이 있는 경우 벡터가 연속된 메모리를 잡는 점을 이용해 파일에서 벡터크기만큼 읽어와 넣어줬습니다. 대신에 이 방식을 이용하면 파일에 출력되었던 벡터의 크기와 현재 입력받으려는 벡터의 크기가 같아야한다는 단점은 있었습니다. 추가로 resize로 크기를 늘리기 위해 RankingData 클래스에 기본생성자를 만들어야 했습니다. 이미 오버로딩된 생성자가 있어 컴파일러가 만들어주지 않았기 때문입니다. 파일이 없다면 초기값 정해 값을 넣어줬습니다. 값은 노말분포로 생성하였고 수업시간에 한대로 1을 더해 우선 0~2사이 범위로 만든뒤 내가 원하는 최대값의 절반을 곱해 점수를 얻었습니다. 노멀분포의 평균은 0 , 표준편차는 0.5로 해서 가우시안분포가 평균에만 몰리지 않고 좀 완만하게 했습니다. 초기데이터를 생성할 때 reserve할 때와 안 할때의 속도차이를 비교해봤는데 reserve할 때가 평균적으로 0.3초 더 빨랐습니다. RankingData클래스에는 동적할당된 변수가 없어 이동생성자를 만들면 오히려 속도가 느려지는 결과가 나왔습니다. 초기데이터를 만든 뒤 sort함수를 써서 정렬을 수행하였습니다. sort와 qsort의 수행 시간 차이는 100000만명이라는 데이터가 너무 적고 클래스의 구조가 너무 단순해서 큰 속도 차이는 안 나왔습니다. sort함수는 내부적으로 대입연산자 오버로딩이 필요했습니다. 제가 이동생성자만 만들고 대입연산자 오버로딩을 안하자 컴파일러 오류가 발생했습니다. 이동생성자를 만든 시점부터 컴파일러가 자동으로 대입연산자를 안 만들어줬기 때문입니다. 하지만 제가 대입연산자를 오버로딩해서 만들어주면 그게 복사건 이동이건 sort함수의 속도가 release일때는 1초, debug일때는 3초내외로 느려졌습니다. 교수님께 질문해보니 아마 컴파일러가 자동으로 생성해주는 대입연산자 오버로딩이 기계어수준에서 더 최적화가 잘되있을거라고 답변해주셨습니다. 그리고 지금처럼 클래스에 따로 복사생성자와 이동생성자를 생성해줄 필요가 없을때는 컴파일러가 자동으로 생성하게 두는 편이 좋다고 하셨습니다. 순위를 세는 것은 교수님이 하셨던 것을 응용해서 했습니다. 점수가 같은 경우 순위를 같게 해야했기 때문에 같은 점수일 때는 증가하지 않는 rank변수와 무조건 증가하는 totalrank 변수로 구분했습니다. 순위가 같은경우에는 rank를 벡터의 원소에 넣고 rank를 증가시키지 않았습니다. totalrank는 전체 얼마나 진행되었는지 알아야했기 때문에 꾸준히 증가시켰습니다. 서로 점수가 다른 경우가 되면 그 때 rank를 totalrank로 갱신하고 벡터의 원소에 rank를 넣어주면 현재 등수는 공동3등이 3명이었다 치면 6등이 되는 식으로 계산됩니다. 10000명씩 플레이하고 그 결과를 보여주는 것도 초기화할때와 똑같은 방식으로 점수를 저장하고 순위를 구했습니다. 대신 점수를 저장하기 전에 예전점수보다 큰지 비교하는 부분이 달랐습니다. 랜덤한 유저에 접근하는 방법은 유니폼 디스트리뷰션으로 얻은 숫자를 인덱스로 사용하는 것으로 해결했습니다. 현재등수와 이전등수를 비교해야하는 부분이 있는데 이건 find\_if함수로 일단 “나”라는 ID를 가진 변수를 벡터에서 찾았습니다. 현재등수를 myrank에 일단 저장해두고RankingData안에 previousRank변수가 0이 아닐때 즉 이전등수가 있을때만 화면에 등수변화를 출력해줬습니다. 나의 윗등수와 아래등수를 표시하는부분은 어려웠던 점에서 설명하겠습니다. 매번 수행시간을 측정하였고 release모드로 하였을때 0.3초 내외로 수행되었습니다. 더이상 플레이하지 않는다면 바이너리 파일로 출력했습니다. 입력받을 때와 똑같이 벡터의 크기만큼 메모리를 읽어와 한번에 쓰는방식으로 수행하였습니다.

**2.어려웠던 점, 아쉬웠던 점**

어려웠던 점은 먼저 파일 입출력이었습니다. 교수님이 알려준 벡터를 한번에 출력하고 입력하는 방식으로 수행하고 싶었는데 벡터의 메모리크기를 아무리 잘 넘겨줘도 오류가 발생했습니다. copy함수를 이용해 문자열형태로 만든 후에 출력해도 입력할때 똑같은 문제가 발생하였습니다. 이유는 입력받을 벡터의 메모리를 확보해놓지 않았기 때문이었습니다. 정확히는 reserve로 확보는 해놨지만 안에 실제 원소를 추가하지 않은 상태여서 발생하는 문제였습니다. resize로 실제 원소까지 추가하자 정상적으로 작동했습니다. 다음으로 어려웠던 부분은 나의 이전등수와 이후등수를 출력하는 부분이었습니다. Find\_if함수의 반환값을 살펴보니 벡터의 반복자를 반환하는 것을 알았습니다. 그래서 반복자를 증가시켜서 다음등수를 찾고 감소시켜서 이전등수를 찾으려고 했는데 이상하게 ‘나’가 출력되지 않고 다른 플레이어가 출력되었습니다. 이유는 제가 player라는 변수에 find\_if한 시점이 sort하기 전이었기 때문이었습니다. 그래서 반복자가 가리키는 주소는 정렬된 시점에 다른 원소가 위치했기 때문에 발생하는 문제였습니다. sort이후에 find\_if함수를 쓰는 것으로 해결했습니다. 아쉬웠던 점은 플레이어 ID를 사람이름처럼 만들지 못한 점이었습니다. 가나다를 잘 조합해서 이름을 만들어보려했지만 이미 다른 모든 코드를 완성한 상태라 만질수록 코드가 망가져 결국 구현하지 못하고 숫자로 구분되는 ID로 만족해야 했습니다. 데이터 초기화 코드와 10000명 플레이하는 코드가 거의 비슷해 함수로 만들려고 했지만 10000명 플레이하는 코드가 인덱스로 벡터에 접근한다는 점 하나가 달라서 코드가 상당부분 반복된 점도 아쉬웠습니다.

**3.느낀점**

과제를 받을 때 구현해야되는 내용도 많고 조건도 까다로워서 되게 막막했습니다. 하지만 교수님이 강의하시는 내용과 겹치는 부분이 많아 수업을 들을수록 문제가 쉽게 풀렸습니다. 과제를 하면서 제일 크게 느낀건 STL이 정말 강력한 라이브러리라는 것이었습니다. vector에는 어떤 거든 넣을 수 있었고 반복자가 익숙해지자 지우고 편집하는 것도 상당히 쉬웠졌습니다. 크기를 유동적으로 할 수 있다는 점과 사용자가 얼마나 잘 쓰느냐에 따라 성능차이가 많이 난다는 점도 상당한 장점이었습니다. 과제를 하면서 제일 잘 했던 점은 시간을 재면서 성능을 체크했던 점이었습니다. emplace와 push, sort와 qsort 등 다양한 상황에서 시간을 체크하고 그 부분을 주석으로 남겨놓았던 게 보고서를 쓸 때도 큰 도움이 되었습니다. 수업시간에 설명하신 무효화된 반복자 문제도 제 프로그램에서 실제로 확인할 수 있었고 람다함수도 처음엔 낮설었지만 find\_if, sort 같은 STL함수들을 많이 사용해 보면서 어느정도 익숙해졌습니다. 벡터를 출력하고 이걸 또 한 번에 입력받는 코드는 여태까지 썼던 파일입출력코드 중 가장 짧고 간단했습니다. copy를 이용해 벡터를 문자열형태로 바꾸는 코드는 나중에 코드를 작성할 때 유용하게 사용될 것입니다.